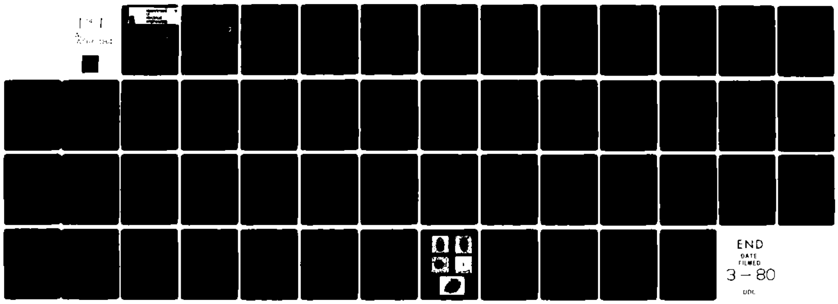
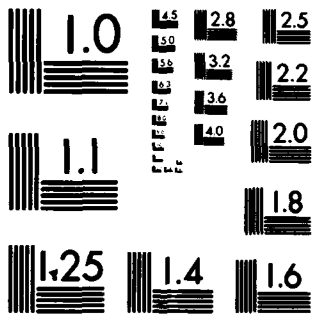
AD-A081 484 COLORADO STATE UNIV FORT COLLINS DEPT OF ELECTRICAL --ETC F/G 12/1
IMPLEMENTATION, INTERPRETATION AND ANALYSIS OF A SUBOPTIMAL BOU--ETC(U)
JAN 80 H ELLIOTT, D B COOPER, F COHEN N00014-75-C-0518
UNCLASSIFIED CSU-TR-JA80-DELENG-1 NL



END
DATE
FILMED
3 – 80
DDC



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU

COLORADO STATE
UNIVERSITY
FORT COLLINS, COLORADO
80523

department of electrical engineering

Implementation, Interpretation and Analysis of a Suboptimal Boundary Finding Algorithm

by

H. Elliott

D. B. Cooper

F. Cohen

P. Symosek

Technical Report #JA80-DELENG-1

January 1980

6) Implementation, Interpretation and Analysis

of a Suboptimal Boundary Finding Algorithm

9) Technical rept.,

10) by

H. Elliott[†]

D. B. Cooper[††]

F. Cohen[††]

P. Symosek[††]

12) 32

Issued as

Colorado State University Technical Report #JA80-DELENG-1

Brown University Technical Report #ENG PRMI 80-1

11) January 1980

14) CSU-TR-JA80-DELENG-1

15) This work was supported in part by the Office of Naval Research under ~~Grant~~ N00014-75-C-0518, in part by the U.S. Army Research Office under Grant DAAG29-78-G-0124, and in part by the National Science Foundation under Grant ENG77-26788.

[†]Department of Electrical Engineering, Colorado State University, Fort Collins, CO 80523

[††]Division of Engineering, Brown University, Providence, RI 02912

406437 [illegible]

Abstract

This report presents a suboptimal boundary estimation algorithm for noisy images which is based upon an optimal maximum likelihood problem formulation. Both the maximum likelihood formulation and the resulting algorithm are described in detail, and computational results are given. In addition, the potential power of the likelihood formulation is demonstrated through the presentation of three simple but insightful analyses of algorithm performance.

Keywords

Image boundary estimation, likelihood maximization, Markov processes, tree searching algorithms.

IMPLEMENTATION, INTERPRETATION AND ANALYSIS OF A SUBOPTIMAL BOUNDARY FINDING ALGORITHM

I. Introduction

In this paper we present an algorithm for estimating object boundaries in noisy black and white images. The algorithm is based upon a maximum likelihood Markov process state estimation formulation developed by Cooper [1], [2]. The images treated consist of a constant grey level object surrounded by a constant grey level background. The entire picture is assumed corrupted by an additive white Gaussian noise field. The algorithm, which sequentially maximizes a suboptimal likelihood function to obtain a boundary estimate consisting of a sequence of pixel edge elements, serves to illustrate the tradeoffs between computational feasibility and theoretical optimality. As is demonstrated, the likelihood formulation also provides a framework for analysis of algorithm performance and for comparison with other algorithms.

Our algorithm was originally motivated by a similar heuristic tree searching algorithm introduced by Martelli [3]. This algorithm estimates boundaries by minimizing a cost function containing gradient and curvature components. It will be shown that this algorithm which is not based upon any optimal problem formulation has potentially poorer performance characteristics. Actually, our approach is more analogous to estimation procedures used in control, communication and information theoretic applications In particular, we develop a Markovian boundary generation model, and view boundary estimation as a problem of estimating the state of this model from noise corrupted measurements.

As discussed by Forney [4], classical problems such as convolutional coding, frequency shift keying, and text recognition can be formulated in the Markov state estimation framework. Scharf and Elliott [5] have also discussed a number of problems in signal and image processing which can be solved using such techniques. A common feature of these algorithms is the formulation of a likelihood function which can be recursively defined and hence sequentially maximized. Furthermore maximization of this likelihood leads to a maximum *a posteriori* probability (MAP) estimate of the state sequence. In many cases elegant dynamic programming (sometimes referred to as the Viterbi algorithm in the information theory literature) algorithms exist for the sequential maximization. In other cases where the state space is large other graph searching algorithms are necessary. As we show below, although it is a simple matter to derive a likelihood function for MAP boundary estimation, it is not possible to calculate this likelihood recursively. The reason for this is the MAP formulation requires use of all the picture data. When object boundaries are estimated sequentially one cannot classify all picture based upon a partial boundary. As a consequence we derive a suboptimal recursive likelihood function. It is based upon the optimal, but it makes use of only picture data in a swath about the boundary. Furthermore due to the size of the state space we are forced to use a suboptimum modified A* (or branch and bound) graph search algorithm to perform the maximization [6]. The algorithm that results is very robust to signal to noise ratios of 2, and although it is more sensitive to model parameter choice it performs well to signal to noise ratios of 1.

A recent paper by Nahi and Lopez-Mora [7] discusses an alternative and very effective probabilistic formulation for sequential boundary estimation. In this approach the likelihood of the picture data is maximized on an individual row by row basis. Since no attempt is made to maximize the joint data likelihood for all of the rows it is also suboptimal. Nor on the surface does it appear that there is an easy way to modify the approach in order to make optimum use of the data. Other potential drawbacks of this approach are that it does not constrain boundary estimates to be continuous and hence it produces jagged boundaries, and its one dimensional nature limits the class of objects to which it is applicable.

The paper is organized as follows. In Section II we present a general maximum likelihood formulation for optimal boundary estimation. With this formulation as a guide, in Section III we present a suboptimal but computationally tractable algorithm for actual boundary estimation. Two alternative Markovian boundary generation models are presented for incorporation into the algorithm. Section IV demonstrates the value of the likelihood formulation for investigating algorithm performance. The suboptimal algorithm is compared with both the Martelli algorithm and the optimal algorithm derived in Section II. We also discuss algorithm performance in the presence of artifacts which are inconsistent with the boundary model. Examples of algorithm performance are given in Section V and some additional comments and concluding remarks are made in Section VI.

II. A Maximum Likelihood Formulation for Boundary Estimation

In this section we present the theoretical framework on which our suboptimal algorithm is based. In particular we formulate the boundary estimation problem in terms of maximizing the joint log likelihood of an hypothesized object boundary and all of the picture data where the likelihood is derived by making use of a probabilistic data generation model.

To begin, a digitalized image will be represented by a picture function g_{ij} whose value corresponds to the grey level of pixel (i,j). We model the picture function g_{ij} as consisting of two components--a true picture component b_{ij}, and a noise component n_{ij}, so that $g_{ij} = b_{ij} + n_{ij}$. The picture is assumed to contain a single object of grey level r_{in} lying in a background of grey level r_{out} where $r_{in} - r_{out} = \Delta > 0$. For convenience we assume an appropriate constant has been subtracted from the original picture function so that $r_{in} = \frac{\Delta}{2}$ and $r_{out} = -\frac{\Delta}{2}$. Therefore b_{ij} only takes on the values $\frac{\Delta}{2}$ and $-\frac{\Delta}{2}$ depending upon whether pixel (i,j) lies inside or outside the object. The noise terms n_{ij} are assumed to be independent, identically distributed (i.i.d.) Gaussian random variables with zero mean and known variance, σ^2.

If an edge element is defined as the line segment separating two adjacent pixels, then an object boundary will consist of a closed directed sequence of edge elements which does not intersect itself, and will be denoted as $\{t_i\}_1^N$. We make the convention that a boundary $\{t_i\}_1^N$ is generated by moving in the clockwise direction around an object. As depicted in Figures 1a and 1b, each directed boundary edge element t_k, $1 \leq k \leq N$, can be uniquely described by a threetuple (i,j,d) where i and j correspond to the coordinates of an adjacent pixel, and d to one of the four possible edge directions. We can then view the unknown edge sequence $\{t_i\}_1^N$ as a discretely indexed vector valued stochastic process with discrete range space. We model this as a Markov process of order K with states $x_i = \{t_j\}_i^{i+K-1}$ and known transition probabilities $P_B(x_i|x_{i-1})$. Appropriate choices for $P_B(\cdot)$ are given in the section to follow.

Let us next derive an expression for the joint likelihood L of an hypothesized boundary edge sequence and the entire picture function. The joint likelihood L is the product of the likelihood L_B of the hypothesized boundary edge sequence, and the likelihood $L_{P|B}$ of the picture function conditioned on the hypothesized boundary. Hence

$$\ln L = \ln L_B + \ln L_{P|B} \quad . \tag{1}$$

Using the Markov chain model we obtain

$$\ln L_B = \ln P_S(x_1) + \sum_{i=2}^{N} \ln P_B(x_i|x_{i-1}) + \ln P_L(N) \tag{2}$$

where $P_S(x_1)$ is the *a priori* probability of a starting state x_1 and $P_L(N)$ is the *a priori* probability of a boundary being of length N. Given a specific boundary edge sequence each pixel corresponds to either object or background. The Gaussian noise model then implies that

$$\ln L_{P|B} = C_1 - \frac{1}{2\sigma^2} \sum_{\substack{\text{pixel } (i,j) \\ \text{in object}}} (g_{ij} - \frac{\Delta}{2})^2 - \frac{1}{2\sigma^2} \sum_{\substack{\text{pixel } (i,j) \\ \text{in} \\ \text{background}}} (g_{ij} + \frac{\Delta}{2})^2 \tag{3}$$

where C_1 is a constant independent of the choice of an hypothesized boundary. Substituting (2) and (3) into (1) yields the following expression for the joint log likelihood

$$\ell nL = \ell nP_S(x_1) + \sum_{i=2}^{N} \ell nP_B(x_i|x_{i-1}) + \ell nP_L(N) + C_1 - \frac{1}{2\sigma^2} \sum_{\substack{\text{in} \\ \text{object}}} (g_{ij} - \frac{\Delta}{2})^2 - \frac{1}{2\sigma^2} \sum_{\substack{\text{in} \\ \text{background}}} (g_{ij} + \frac{\Delta}{2})^2 \tag{4}$$

Note that (4) simplifies to

$$\ell nL = \ell nP_S(x_1) + \sum_{i=2}^{N} \ell nP_B(x_i|x_{i-1}) + \ell nP_L(N) + C_2 + \sum_{\substack{\text{in} \\ \text{object}}} (\Delta/2\sigma)(g_{ij}/\sigma) - \sum_{\substack{\text{in} \\ \text{background}}} (\Delta/2\sigma)(g_{ij}/\sigma) \tag{5}$$

where C_2 is independent of the choice of an hypothesized boundary. Hence, although (4) appears to be quadratic in the picture function g_{ij}, when all the picture data is used, only the linear correlation terms effect any maximization.

An optimal boundary estimation algorithm would maximize (5) over all possible boundary edge sequences. This would produce a maximum *a posteriori* probability (MAP) estimate of the state sequence $\{x_i\}$ and hence the boundary edge sequence $\{t_i\}$. However, since (5) makes use of all the picture data, it is computationally impractical to maximize it. Furthermore, since computation of the two summations involving the picture function g_{ij} requires knowledge of the complete hypothesized boundary, (5) cannot be sequentially maximized. In view of these problems a suboptimal boundary estimation algorithm is presented in the following section.

III. A Suboptimal Boundary Estimation Algorithm

In this section we present a suboptimal boundary estimation algorithm which is based upon maximizing a likelihood function similar to (5), but one which can be maximized using sequential procedures such as the A* (branch and bound) algorithm [6]. The basic approach involves limiting the use of picture function data to a small region about an hypothesized boundary.

As pointed out by Martelli [3], sequential boundary estimation can be viewed as finding a path through a directed graph. For our formulation each node of the graph corresponds to a K-dimensional Markov state defined by the boundary generation model introduced in the previous section. A likelihood value is assigned to each node, and it corresponds to the maximum of the likelihoods for all paths leading to that node from a predetermined start node. As shown in Fig. 2, each node has exactly three successor nodes corresponding to the three states which can be obtained by adding a new edge element to an hypothesized boundary edge sequence.

By defining a goal node to be a boundary state lying within a neighborhood of the start node, as depicted in Fig. 3, graph theoretic search procedures such as the A* algorithm can be applied to obtain a maximum likelihood path from the start node to a goal node. The states along such a path would then define the boundary estimate. These algorithms generate only the portions of the graph needed to continue the search. For example, the basic step in the procedure involves searching the present graph for the node with the largest likelihood, adding to the graph any of its three successors not already on the graph, and recalculating the likelihood of any of the successor nodes which are already on the graph. As a result, it is computationally desirable to recursively calculate the likelihood of a successor state from knowledge of the likelihood of its predecessor. In view of this, if x_i is an arbitrary graph node and x^j_{i+1}, j = 1,2,3 are its three successor states as depicted in Fig. 2, then the following suboptimal log likelihood function can be formulated based upon the optimal likelihood (5),

$$\ell n\tilde{L}(x^j_{i+1}) = \ell n\tilde{L}(x) + \ell nP_B(x^j_{i+1}|x) + D(x^j_{i+1}) \qquad j=1,2,3 \quad . \tag{6}$$

$D(x^j_{i+1})$ is the change in picture data likelihood caused by adding node x^j_{i+1} to the boundary sequence defined by the most likely path from the start node to node x^j_{i+1}, and $P_B(x^j_{i+1}|x_i)$ is the state transition probability defined in the previous section. Methods for calculating these two quantities are discussed in detail below.

Calculating Picture Function Contribution $D(x^j_{i+1})$

Picture data is incorporated into (6) by using pixels contained in a swath about an hypothesized edge sequence. Figure 4 gives examples of data swaths for alternative boundary edge sequences. This use of picture data is suboptimal since for the situations depicted in Figure 4, the log likelihoods of the two alternative paths would be evaluated using different picture data. As will be discussed in more detail in the section to follow, optimal use of the data would require comparison of these two alternative boundary paths using likelihoods which incorporated the same picture data. Our data usage is globally suboptimum, but it is locally optimum. The global suboptimality is pertinent to the relative frequency with which the sequential estimator leaves the vicinity of the true path and must backtrack. The local optimality is pertinent to the relative frequency with which the estimate will be in error by a small number of pixels, e.g., one or two pixels. Since the latter accuracy consideration may be very important in some applications we discuss it further below.

$D(x_{i+1}^{j})$ is calculated by making use of picture data contained in a (4x4) pixel block surrounding the last edge of the boundary edge sequence leading to node x_i. The purpose of this (4x4) template is to optimally use data in the region of the deepest edge element of a path in order to choose among the following three edge elements and to extend the path in an optimal way. This has the advantage of permitting optimal detection for short but arbitrarily curvy boundary subsequences whereas Hueckle type operators [9], for example, are restricted to straight lines.

As shown in Fig. 5, 15 possible extension edge sequences out of the (4x4) block are considered, 5 for each of the successor states x_{i+1}^{j}, $j=1,2,3$. Assuming a boundary edge sequence to be generated by moving in the clockwise direction about an object, the original sequence and any of the 15 extension paths uniquely classify each of the 16 pixels in the block as to being either inside or outside the object. Based upon (4) and (5) we have experimented with and anlyzed two alternative approaches for assigning likelihoods to each classified pixel. In view of (4) a natural, and our first choice involved assigning the quadratic values

$$-\frac{1}{2}\ \ell n(2\pi\sigma^2) - \frac{1}{2\sigma^2}(g_{ij} - \frac{\Delta}{2}) \tag{7a}$$

or

$$-\frac{1}{2}\ \ell n(2\pi\sigma^2) - \frac{1}{2\sigma^2}(g_{ij} + \frac{\Delta}{2}) \tag{7b}$$

depending upon whether pixel (i,j) was classified as in object or background respectively. However, as we discuss in the Section IV to follow, improved algorithm performance is obtained by assigning values more analogous to the linear data terms in (5). In particular we assign the values

$$-(\Delta/2\sigma)^2 + (\Delta/2\sigma)(g_{ij}/\sigma) \qquad (8a)$$

or

$$-(\Delta/2\sigma)^2 - (\Delta/2\sigma)(g_{ij}/\sigma) \qquad (8b)$$

depending upon whether pixel (i,j) is classified as in object or background respectively.

For each node x^j_{i+1} five values $D_k(x^j_{i+1})$, $k=1,2,\ldots,5$ are calculated; one for each of the five extension paths. Observe that these calculations can be performed in a computationally efficient manner. It should be apparent from Figure 5 that for fixed j, each (4x4) template, k, can be obtained from template, k=1, by reclassifying a single pixel. Hence $D_k(x^j_{i+1})$ can be obtained from $D_{k-1}(x^j_{i+1})$ by adding an appropriate constant. If any extension path causes a loop with the optimal edge sequence leading to state x_i then the corresponding $D_k(^j_{i+1})$ is assigned a value of $-\infty$. It should also be noted that some of the data in the (4x4) template was used in calculating $\tilde{L}(x_i)$. Hence only additions and changes are incorporated into $D_k(x^j_{i+1})$. This implies that the complete data swath for a boundary edge sequence consists of the union of a sequence of these (4x4) blocks. Finally we define

$$D(x^j_{i+1}) = \max_{1\leq k\leq 5} D_k(x^j_{i+1}) \quad . \qquad (9)$$

Though the required computation in calculating the 15 values of $D_k(x^j_{i+1})$ is seemingly large, it is in fact modest since, as pointed out above, these values can be calculated recursively. We consider this algorithm to be a very important part of this approach since the large number of template matchings required might otherwise be prohibitive.

Calculating Boundary Contribution $\ln P_B(x_{i+1}|x_i)$

We have experimented with two methods for calculating the Markovian transition probabilities $P_B(x^j_{i+1}|x_i)$. To make some initial comparisons with results previously presented in the literature, we first chose $P_B(x^j_{i+1}|x_j)$ so that $-\ln P_B(x^j_{i+1}|x_i)$ roughly resembled the Martelli curvature cost [3]. We have also developed a method for calculating transition probabilities based upon a dynamic boundary generation model and implementation of a Kalman filter.

The Martelli curvature constraint [3] was based upon the assumption that the boundaries of interest were locally smooth and of low curvature. Under these assumptions we can define our state dimension K=8, and

$$P_B(x^j_{i+1}|x_i) = Ae^{f(\theta_j)} \tag{10a}$$

$$A = \sum_{j=1}^{3} e^{f(\theta_j)} \tag{10b}$$

$$f(\theta_j) = -a|\theta_j| - b\theta_j^2 \tag{10c}$$

where a>0, b>0, are arbitrary constants, and θ_j is the angle depicted in Figure 2. A is a normalization factor which assures that $\sum_{j=1}^{3} P_B(x^j_{i+1}|x_i)=1$. Since $\ln P_B(x^j_{i+1}|x_i) = \ln A + f(\theta_j)$ has a maximum for $\theta_j=0$ these transition probabilities favor straight boundaries and discourage sharp directional changes.

In implementing this scheme we observed that for coursely quantized images θ_j varies considerably along constant curvature paths such as circular arcs. Hence to reduce the variability in our measurement of θ_j we have developed a look-up table procedure for determining directions of line segments passing through sequences of four edges such as shown in Figure 2. The directions of the line segments for the first four edges and for the second four edges are then compared to determine θ_j.

The Kalman filtering approach requires additional *a priori* information about boundary shape. It is well known that discrete "time" linear dynamic systems driven by sequences of i.i.d. random variables represent Markov processes. These systems can provide good models for many types of smooth object boundaries. Here we will assume that the boundaries of interest look like perturbed or distorted ellipses. This model is valid for a number of different human organ boundaries in CAT scans, or conventional tomographs, and also the cloud shown in Figure 10e.

Using a Euclidean coordinate system, let a sequence of points on the true object boundary be denoted by the vectors v(k) (2x1), $1 \leq k \leq M$. Having restricted the image boundary to consist of a sequence pixel edges t_i, $1 \leq i \leq N$, points lying on these edges represent quantized versions of the points v(k), and will be denoted as $v_q(k)$, $1 \leq k \leq M$. Since elliptical trajectories can be generated as solutions to second order systems, an appropriate model for distorted ellipses would be

$$v(k+1) = Av(k) + b + u_k \tag{11}$$

$$v_q(k) = v(k) + w_k \quad , \tag{12}$$

where A (2x2) and b (2x1) are constant matrices, and u_k(2x1) and w_k(2x1) are sequences of zero mean, i.i.d., Gaussian random vectors with covariance matrices $\sigma_u^2 I$ and $\sigma_w^2 I$ respectively. The process u_k can be viewed as the distortion mechanism while the process w_k approximates the quantization error. We also assume u_k and w_k to be uncorrelated. For an ellipse consisting of M points, having a ratio of major to minor axis of ρ, a rotation angle of θ, and a center at c(2x1), appropriate choices for A and b would be

$$A = \begin{bmatrix} \gamma + \delta(\frac{1-\rho^2}{\rho})\sin\theta\cos\theta & \delta(\frac{1}{\rho}\cos^2\theta+\rho\sin^2\theta) \\ -\delta(\rho\cos^2\theta+\frac{1}{\rho}\sin^2\theta) & \gamma-\delta(\frac{1-\rho^2}{\rho})\sin\theta\cos\theta \end{bmatrix}$$

$$b = [I-A]c$$

$$\gamma = \cos\frac{2\pi}{M}$$

$$\delta = \sin\frac{2\pi}{M} \quad .$$

From knowledge of A, b, σ_u^2, and σ_w^2 one can construct the well known steady state Kalman filter (or predictor) for generating an estimate $\hat{v}(k)$ of $v(k)$:

$$\hat{v}(k+1) = A\hat{v}(k) + b\, G[v_q(k) - \hat{v}(k)] \qquad (13)$$

$$G = AP[\sigma_w^2 I+P]^{-1} \qquad (14)$$

$$P = APA^T + \sigma_u^2 I + AP[\sigma_w^2 I+P]^{-1}PA^T \quad . \qquad (15)$$

To see how this filter can be used for generating the transition probabilities $P_B(x_{i+1}^j|x_i)$ consider Figure 6. Figure 6 shows three possibilities $v_q^1(k)$, $v_q^2(k)$, and $v_q^3(k)$ for the point $v_q(k)$. These correspond to the end points of the three alternatives, t_{i+1}^1, t_{i+1}^2, t_{i+1}^3, for edge element t_{i+1} given t_i. The estimate $\hat{v}(k)$ of $v(k)$ is also shown. Having assumed u_k and w_k to be Gaussian, one can show that if $v_q(k)$ were unconstrained then it would be Gaussian, and conditioned on $v_q(k-1)$ it would have mean $\hat{v}(k)$ and covariance $P + \sigma_w^2 I$. However, since $v_q(k)$ is constrained to lie on a pixel edge we can discretize the Gaussian density function to obtain approximate transition probabilities.

Let the Markov process state x_i be the single edge element t_i, then

$$P_B(x_{i+1}^j|x_i) = P(v_q^j(k+1)|v_q(k))$$

$$\tilde{=} P_G(v_q^j(k),\hat{v}(k),\sigma_w^2 I+P)/\sum_{j=1}^{3} P_G(v_q^j(k),\hat{v}(k),\sigma_w^2 I+P) \quad , \qquad (16)$$

where

$$P_G(x,m,\Sigma) = (2\pi|\Sigma|)^{-\frac{1}{2}}\exp(-\frac{1}{2}(x-m)^T\Sigma^{-1}(x-m)) \qquad (17)$$

In implementing this scheme, since the dynamic system does not generate equidistant point, we have found it helpful to choose M=3N so that on the average we generate 3 estimates per pixel. To obtain the value $\hat{v}(k)$ used in (17) the predictor is run sequentially from $\hat{v}(k-1)$ until a pixel boundary is crossed.

<u>Graph Searching</u>

The state likelihoods calculated by (6) are used in conjunction with a modified A* algorithm to find the boundary estimate. First since our (4x4) data template was developed under the hypothesis that the boundary passed through the template, before we expand a state, i.e. add its successor states to the graph we test this hypothesis. In particular, if the probability that the (4x4) template (centered about the last edge element of the state to be expanded) is completely inside the object or background is above a fixed threshold then this state is not expanded. Second, to make the algorithm computationally feasible in terms of both computer memory and execution time requirements it is necessary to periodically prune the graph of nodes which had little probability of being on a boundary. If addition of a set of successor states x^j_{i+1}, j=1,2,3 increased the maximum depth of the graph, then all graph branches a distance (T+1) or more back from the states x^j_{i+1} are pruned from the graph provided they do not lie on the most likely path from the starting state to states x^j_{i+1}.

In Section V to follow we present some examples to illustrate the algorithms performance.

IV. Interpretation and Analysis

In this section we show how the mathematical framework introduced in Section II can be used to do some simple but insightful analyses of the boundary estimation algorithm outlined in Section III. The first analysis compares the operation of this algorithm with that of the Martelli algorithm, while the second examines the effects of the algorithms restricted use of picture function data and shows that performance will not be significantly impaired. Finally, the third analysis shows how it is possible to predict algorithm performance when certain unexpected artifacts are encountered.

Comparison with Martelli Algorithm

If the first method for calculating transition probabilities, using local curvature information, is incorporated into the algorithm, the essential difference between it and the Martelli algorithm is the method in which picture function data is employed for estimation. The Martelli algorithm searches a directed graph for a minimum cost path where one component of the nodal cost function is a directional derivative calculated from local picture function data. This is somewhat analogous to the picture function component of (6) obtained by classifying data in the (4x4) template. In this subsection we compare the performance of the two algorithms by looking at pairs of edge sequences (paths). One path is assumed to lie on the true boundary while the other is assumed to lie in the object or background. We then calculate an error probability for each algorithm, that is the probability that the path lying in the object or background is more likely or of least cost. We first consider the case of a single erroneous edge and show that our new algorithm has a lower probability of initially leaving a correct path. The implication of this is that fewer nodes will be added to the graph. We next look at longer paths where one path is straight and the other curvy. Here it is shown that our algorithm performs better when the true boundary is curvy while Martelli's algorithm performs better when the true boundary is straight. In fact, in the latter case, the Martelli algorithm performs better than the optimal algorithm which maximizes (5). This can be explained, however, by realizing that the optimal formulation assumes no *a priori* information about boundary curvature while the Martelli algorithm is designed to favor straight paths.

First consider the patch of picture in Figure 7. It shows a correct edge sequence (lying on the object boundary), $\{t_i\}_1^5$, along with an erroneous edge t_3'. Let us calculate the probability of each algorithm expanding (generating successor states to) the incorrect state $x'=\{\ldots,t_1,t_2,t_3'\}$ rather than the correct state $x=\{\ldots,t_1,t_2,t_3\}$. Assuming the decision is based solely on use of picture function data, Martelli's algorithm would decide x' rather than x if the data cost $c(x')$ were less than the cost $c(x)$. The cost function used by Martelli approximates the directional derivative across an edge and makes use of 4 pixels in the row (or column) containing the edge (2 on each side). In addition, the constant 2Δ is added to insure that the cost is zero if calculated across a true boundary in the noise free case. Using this function we obtain

$$c(x') = 2\Delta + g_{21} + g_{22} - g_{23} - g_{24}$$

$$c(x) = 2\Delta + g_{13} + g_{23} - g_{33} - g_{43}$$

Therefore

$$P(\text{deciding } x'|x \text{ correct})$$

$$= P(c(x)-c(x') > 0|x \text{ correct})$$

$$= P(2g_{23} + g_{13} + g_{24} - g_{21} - g_{22} - g_{33} - g_{43} > 0|x \text{ correct}$$

$$\equiv P(G > 0|x \text{ correct})$$

Conditioned on x being correct, G is a normally distributed random variable with mean $-\Delta$ and variance $10\sigma^2$, i.e., $G \sim N(-\Delta, 10\sigma^2)$. This implies that

$$P(G>0|x \text{ correct}) = \int_{\frac{1}{\sqrt{10}}(\frac{\Delta}{\sigma})}^{\infty} (2\pi)^{-\frac{1}{2}} \exp(-s^2/2)ds \quad . \tag{18}$$

Using our algorithm calculation of the data likelihood for both x and x′ involves classifying the same 16 pixels shown in Figure 7. The only difference between the two likelihoods is the value assigned pixel (2,3) for the likelihood $\ell n\tilde{L}(x)$ it is assigned $-(\Lambda/2\sigma)^2 - (\Lambda/2\sigma)(g_{23}/\sigma)$ while for $\ell n\tilde{L}(x')$ it is assigned $-(\Lambda/2\sigma)^2 + (\Lambda/2\sigma)(g_{23}/\sigma)$. Therefore

$$\begin{aligned} P(\text{deciding } x'|x \text{ correct}) &= P(\ell n\tilde{L}(x') - \ell n\tilde{L}(x) > 0|x \text{ correct}) \\ &= P((\Delta/\sigma^2)g_{23} > 0|x \text{ correct}) \\ &= P(H > 0|x \text{ correct}) \end{aligned}$$

Conditioned on x being correct H is $N(-(1/2)(\Lambda/\sigma)^2,(\Lambda/\sigma)^2)$ and

$$P(H>0|x \text{ correct}) = \int_{\frac{1}{2}(\frac{\Delta}{\sigma})}^{\infty} (2\pi)^{-\frac{1}{2}}\exp(-s^2/2)ds \quad . \tag{19}$$

Since the lower limit of the integral (19) is 1.6 times that of (18) the latter probability is higher. As a numerical example, consider the case of a signal to noise ratio, (Λ/σ), of one. Then the lower integral limits in (18) and (19) are .5 and .625, and the corresponding error probabilities are .265 and .309 respectively. This gives an indication that this algorithm is less likely to explore extraneous paths than Martelli's, or equivalently, it would return substantially the same performance in the presence or larger noise variance.

Next consider the eight pairs of paths shown in Figure 8. In each case path 1 is straight and path 2 is a curvy perturbation obtained by moving four pixels from inside object to inside background or vice-versa. If paths 1 and 2 have Martelli costs c_1 and c_2 and log likelihoods ℓnL_1

and $\ln\tilde{L}_2$ as defined by our algorithm, then the algorithms choose paths 1 or 2 in accordance with

$$G = (c_1 - c_2) \underset{\text{path 1}}{\overset{\text{path 2}}{\gtrless}} 0 \tag{20a}$$

$$H = (\ln\tilde{L}_2 - \ln\tilde{L}_1) \underset{\text{path 1}}{\overset{\text{path 2}}{\gtrless}} 0 \quad . \tag{20b}$$

G and H will each have two components a boundary statistic denoted as B, either curvature cost or transition probability, and a data statistic denoted as A, either gradient or template. Conditioned on one path being correct (lying on boundary) and the other being incorrect, then only A is a random variable and the total statistic (A+B) is normally distributed with mean and variance which are a function of which path is assumed correct. Hence the probability of an error, that is each algorithm finding the incorrect path of least cost or more likely, is of the form

$$P_\rho = P(A+B \gtrless 0 | \text{paths 1 and 2})$$

$$= \int_\rho^\infty (2\pi)^{-\frac{1}{2}} \exp(-s^2/2)ds \tag{21}$$

where

$$\rho = \frac{\mp(B+\text{Mean}(A))}{\text{Standard deviation}(A)} \quad . \tag{22}$$

Assuming B=0 and a signal to noise ratio of one, Table 1 gives ρ and P_ρ for each of the eight cases both when the straight path 1 is assumed correct and also when the curvy path 2 is assumed correct. When both possibilities are equally likely then the average probability of an error, P_a, is just the average of P_ρ for the two cases. This is also calculated in Table 1. If an algorithm were employed which chose paths according to the optimal likelihood function (5) then the only pixels entering the error statistic would be the four lying between the two paths. Hence one can easily calculate ρ and P_ρ for this algorithm and this data is given in Table 1 as well. A common property of our algorithm and the optimal is the symmetry of the error statistic. One obtains the same error probability regardless of whether path 1 or path 2 is assumed correct. This is not true for the Martelli algorithm. In particular the Martelli algorithm does considerably better when the correct path is straight. In fact in this case it out performs the optimal. This is because it has been designed to favor straight paths, while neither the data statistic for our algorithm nor the optimal, favors any type of curvature. On the other hand, because of the assymetry of the Martelli statistic our algorithm gives considerably better performance if the correct path is curvy. Although, as expected, it does not perform as well as the optimal. A more detailed comparison of our algorithm and the optimal is given in the next subsection. Finally in comparing overall performance, observe that for five of the cases our algorithm gives significantly lower average error probability while Martelli's algorithm does significantly better in only two of the cases.

Justification for the Suboptimal Boundary Estimator

In Section II we derived a likelihood function (5) for optimal MAP estimation of the state sequence $\{x_i\}$ or equivalently the boundary edge sequence $\{t_i\}$. A key feature of (5) is the way it incorporates picture function data. First, it makes use of all the picture function data. Second, as seen in (4), the data enters quadratically. However, after simplification only the linear terms effect the maximization. To develop a computationally feasible algorithm picture data had to be used in a suboptimal manner. In this subsection we investigate analytically, performance effects caused by our suboptimal data usage. In particular we show that our boundary estimator performs substantially the same as the optimal. We also give justification for using picture data linearly in our suboptimal algorithm rather than quadratically.

As done in the previous subsection, we consider two alternative paths. Path 1 is assumed to lie on the true boundary and we calculate the probability of an error, that is that path 2 appears more likely. Rather than consider specific paths such as in Figure 8, we will consider two more general cases. As shown in Figure 4a, in case 1 we will assume that the two paths are far enough apart so that the two data swaths used by the suboptimal algorithm do not intersect. In case 2, Figure 4b, we assume that the paths are close enough together so that the swath sections to the left of path 2 and to the right of path 1 overlap.

For each case we consider three algorithms. The optimal algorithm based on (5) makes use of all the data in the picture region containing the two paths. However, a more practical approach, although less optimal, is to incorporate only the data in the two swaths, but use all of it in calculating the likelihood for each path. We compare this near optimal algorithm with our algorithm and it will be designated Algorithm 1. The second algorithm considered is the final version of our suboptimal algorithm where picture data enters the likelihood function (6) linearly according to (8). With this algorithm the likelihood of a specific path contains only data from the swath about that path. As mentioned in Section III, since picture data enters equation (4) quadratically, in our early experimental work we chose to incorporate picture data into (6) quadratically according to (7). In order to demonstrate the improved performance obtained by using the linear rather than the quadratic terms, Algorithm 3 is also analyzed. It is the same as Algorithm 2 but incorporates picture data using (7) rather than (8).

Before proceeding some additional notation is needed. Consider Figure 4 and let $S_{\ell m}$, $\ell\varepsilon(1,2)$, $m\varepsilon(L,R)$ denote the pixel subset contained on side m of swath ℓ (swath containing path ℓ). Next let S_ℓ denote the entire set of pixels in swath ℓ so that $S_\ell = S_{\ell L} \cup S_{\ell R}$. Define the number of pixels in set $S_{\ell m}$ or S_ℓ as $N_{\ell m}$ or N_ℓ respectively. Finally for the overlapping case of Figure 4b, let $S_I = S_{1R}\ S_{2L}$ and N_I be the subset and number of pixels in the intersection of S_{1R} and S_{2L}.

Case 1 - Algorithm 1: In this case we assume S_1 and S_2 to be disjoint, and path 1 lies on the true boundary. Algorithm 1 uses likelihood (5) but only for pixel data in S_1 and S_2. Defining L_1 and L_2 as the likelihoods of paths 1 and 2 respectively, then an error is made if $Q_{11} > 0$ where

$$Q_{11} = \ell n L_2 - \ell n L_1 = \sum_{\text{path } 2} \ell n\, P_B(x_i|x_{i-1}) - \sum_{\text{path } 1} \ell n\, P_B(x_i|x_{i-1})$$

$$- \sum_{(i,j)\varepsilon S_{2L}} (\Delta/2\sigma)(g_{ij}/\sigma) - \sum_{(i,j)\varepsilon S_{1R}} (\Delta/2\sigma)(g_{ij}/\sigma) \qquad (23)$$

As in the previous subsection $Q_{11} = A_{11} + B$ where B is the boundary statistic consisting of the transition probability terms in (23), and A_{11} the data statistic consisting of the picture function terms. Conditioned on two specific paths only A_{11} is a random variable. It is Gaussian with distribution

$$A_{11} \sim N(-2(N_{1R}+N_{2L})(\Delta/2\sigma)^2,\ 4(N_{1R}+N_{2L})(\Delta/2\sigma)^2) \quad . \qquad (24)$$

Hence, conditioned on paths 1 and 2 the probability of error

$$P_{11}=P(Q_{11} > 0|\text{paths 1 and 2}) =$$

$$= \int_{\rho_{11}}^{\infty} (2\pi)^{-\frac{1}{2}}\exp(-s^2/2)ds \qquad (25)$$

where

$$\rho_{11} = \frac{-B\text{-Mean}(A_{11})}{\text{Standard deviation}(A_{11})} \qquad (26)$$

For B=0 and $N_{1R}=N_{2L}\overset{\Delta}{=}N$ we obtain

$$\rho_{11} = \frac{1}{\sqrt{2}} (N)^{\frac{1}{2}}(\frac{\Delta}{\sigma}) \quad . \tag{27}$$

Numerical values for ρ_{11} and P_{11} when the signal to noise ratio (Δ/σ) is one and two, and when the number of pixels, N, in swaths S_{2L} and S_{1R} is 4 and 8, are given in Table 2.

<u>Case 1 - Algorithm 2:</u> For our suboptimal algorithm the likelihood associated with path 1 only contains picture data in swath S_1 while that associated with path 2 only contains data from swath 2. Hence using equations (6) and (8), if $\tilde{L}_1$ and $\tilde{L}_2$ are the likelihoods associated with paths 1 and 2 respectively then an error is made if $Q_{12} > 0$ where

$$\begin{aligned} Q_{12} = \ln\tilde{L}_2 - \ln\tilde{L}_1 = & \sum_{\text{path } 2} \ln P_B(x_i|x_{i-1}) - \sum_{\text{path } 1} \ln P_B(x_i|x_{i-1}) \\ & + \sum_{(i,j)\epsilon S_{1L}} (\Delta/2\sigma)(g_{ij}/\sigma) + \sum_{(i,j)\ S_{2R}} (\Delta/2\sigma)(g_{ij}/\sigma) \\ & - \sum_{(i,j)\epsilon S_{1R}} (\Delta/2\sigma)(g_{ij}/\sigma) - \sum_{(i,j)\ S_{2L}} (\Delta/2\sigma)(g_{ij}/\sigma) \\ & + (N_1-N_2)(\Delta/2\sigma)^2 \end{aligned} \tag{28}$$

If $Q_{12} = A_{12} + B$ where B is the boundary statistic and A_{12} is the data statistic, then

$$A_{12} \sim N(-2N_{2L}(\Delta/2\sigma)^2 , (N_1 + N_2)(\Delta/2\sigma)^2) \quad . \tag{29}$$

Again we can calculate an error probability

$$\begin{aligned} P_{12} &= P(Q_{12} > 0|\text{paths 1 and 2}) \\ &= \int_{\rho_{12}}^{\infty} (2\pi)^{-\frac{1}{2}}\exp(-s^2/2)ds \quad . \end{aligned} \tag{30}$$

For B=0 and $N_{1R} = N_{1L} = N_{2R} = N_{2L} \overset{\Delta}{=} N$ we obtain

$$\rho_{12} = \frac{1}{2}(N)^{\frac{1}{2}}(\frac{\Delta}{\sigma}) \quad . \tag{31}$$

Numerical values for ρ_{12} and P_{12} can be found in Table 2. Observe that $\rho_{11} = \sqrt{2}\ \rho_{12}$. Thus our suboptimal data usage results in a boundary estimator which is quite good. In fact for as few as 8 pixels in a swath (N=4), and a signal to noise ratio $(\Delta/\sigma) = 1$, the error probability is only .150. Also, as is apparent from examination of Table 2, if we consider longer paths with more pixels in a swath, or pictures with larger signal to noise ratios, the error probability quickly decreases to zero.

Case 1 - Algorithm 3: If we use an algorithm similar to algorithm 2 but one where data likelihoods are calculated using the quadratic terms (7) rather than the linear terms (8) we obtain an error statistic $Q_{13} = A_{13} + B$ where

$$A_{13} = \sum_{(i,j)\varepsilon S_{1L}} \frac{1}{2\sigma^2}(g_{ij} + \frac{\Delta}{2})^2 + \sum_{(i,j)\varepsilon S_{1R}} \frac{1}{2\sigma^2}(g_{ij} - \frac{\Delta}{2})^2$$
$$- \sum_{(i,j)\varepsilon S_{2L}} \frac{1}{2\sigma^2}(g_{ij} + \frac{\Delta}{2})^2 - \sum_{(i,j)\varepsilon S_{2R}} \frac{1}{2\sigma^2}(g_{ij} - \frac{\Delta}{2})^2$$
$$+ \frac{1}{2}(N_1 - N_2)\ln(2\pi\sigma^2) \qquad (32)$$

Although A_{13} is not Gaussian if N_1 and N_2 are large then it is reasonable to approximate it as Gaussian. Since A_{13} has mean and variance

$$\text{Mean}(A_{13}) = \frac{1}{2}(N_1 - N_2)(1 + \ln(2\pi\sigma^2)) - 2N_{2L}(\Delta/2\sigma)^2$$
$$\text{Var}(A_{13}) = \frac{1}{2}(N_1 + N_2) + 4N_{2L}(\Delta/2\sigma)^2$$

For B=0 and N as defined above

$$\rho_{13} = \frac{1}{2}(N)^{\frac{1}{2}}(\frac{\Delta}{\sigma})\frac{1}{(1+2(\frac{\sigma}{\Delta})^2)^{\frac{1}{2}}} \quad . \qquad (33)$$

Numerical values for ρ_{13}, and the error probability P_{13} under the Gaussian assumption are given in Table 2. Clearly Algorithm 2 outperforms Algorithm 3 for all finite signal to noise ratios, and significant improvement is obtained for signal to noise ratios of one or less.

Case 2 - Algorithm 1: As illustrated in Figure (4b) for this case we assume paths 1 and 2 are closer together so that $S_I = S_{1R}\ S_{2L}$ is not empty. However we do not allow S_1 to intersect S_{2R} or S_2 to intersect S_{1L}. The overlap does not influence the performance of Algorithm 1 hence $\rho_{21} = \rho_{11}$ and the error probability $P_{21} = P_{11}$.

Case 2 - Algorithm 2: Although for algorithm 2 the error statistic, Q_{22}, is still given as in Case 1 by equation (28), the overlapping data in S_{1R} and S_{2L} effect the distribution of the corresponding data statistic A_{22}. The pixels contained in the overlapping set S_I enter A_{22} twice rather than once causing A_{22} to have a larger variance than A_{12}. In particular

$$A_{22} \sim N(-2N_{2L}(\Lambda/2\sigma)^2,\ (N_1+N_2+2N_I)(\Lambda/2\sigma)^2) \qquad (34)$$

This causes a degradation in performance in comparison with both Case 1, Algorithm 2, and Cases 1 and 2, Algorithm 1. For example, if we consider a limiting situation where $S_{1R} = S_{2L}$ so that $N_I = N$ we obtain

$$\rho_{22} = \frac{1}{\sqrt{6}} (N)^{\frac{1}{2}}(\frac{\Lambda}{\sigma}) \qquad . \qquad (35)$$

Hence $\rho_{12} = (4/3)^{\frac{1}{2}} \rho_{22}$, and $\rho_{11} = \rho_{22} = \sqrt{3}\, \rho_{22}$. However, as illustrated in Table 2, performance is still quite good.

Case 2 - Algorithm 3: For Algorithm 3 the data overlap actually causes an improvement in performance. In this case the non-Gaussian data statistic A_{23} has

$$\text{Mean } (A_{23}) = \frac{1}{2}(N_1-N_2)(1+\ln(2\pi\sigma^2)) - 2N_{2L}(\Lambda/2\sigma)^2 \qquad (36)$$

$$\text{Var } (A_{23}) = \frac{1}{2}(N_1+N_2-N_I) + 4N_{2L}(\Lambda/2\sigma)^2 \qquad . \qquad (37)$$

Making the Gaussian approximation and taking $N_I = N$ we obtain

$$\rho_{23} = \frac{1}{2}(N)^{\frac{1}{2}}(\frac{\Lambda}{\sigma}) \frac{1}{(1+\frac{3}{2}(\frac{\sigma}{\Lambda})^2)^{\frac{1}{2}}} \qquad . \qquad (38)$$

Although the overlap causes a degradation in the performance of algorithm 2 and an improvement in that of Algorithm 3, as can be seen from Table 2, for the case of small signal to noise ratios. Algorithm 2 still outperforms Algorithm 3. This is the more important situation since for large signal to noise ratios all error probabilities are small.

<u>Estimating Projections</u>

We would like to be able to predict algorithm performance when the object boundary contains artifacts which are somewhat inconsistent with the boundary generation model. Consider, for example the projection shown in Figure 8, and assume the notation and terminology of the previous analyses. If the state transition probabilities $P_B(\cdot|\cdot)$ are small for paths of high curvature, that is the Markov process boundary model favors smooth straight paths, then the contribution to the boundary statistic B for path 1 will be larger than that for path 2. Thus the statistic B will be positive, and tend to support an erroneous decision in favor of path 2. The only way that the overall decision statistic, A+B, can support a correct decision is for A+B<<0, or A sufficiently negative. This will be the case if N_1 and N_2 are large enough, that is if there is sufficient data so that its contribution to the decision statistic outweighs the boundary model contribution. More specifically we require

$$\rho = \frac{-B - \text{Mean}(\Lambda)}{\text{Standard deviation}(\Lambda)} >> 1 \tag{39}$$

Using (29) we obtain

$$\frac{-B + 2N_{2L}(\Lambda/2\sigma)^2}{(N_1+N_2)\ (\Lambda/2\sigma)} >> 1 \tag{40}$$

If the projection is long relative to the number of edge elements used in calculating transition probabilities for B, B will remain roughly constant as N_1 and N_2 increase. Using (40), and estimating B for a specific boundary model allows one to determine roughly the required swath width for correct estimation. A more accurate determination of the estimators ability to track around projections requires a more careful analysis of the sequential behavior of the algorithm. An example illustrating the results of applying our algorithm to a projection such as in Figure 9 is given in the following section.

An alternative approach to handling boundaries with unusual artifacts such as occasional projections or others, is to treat these as pattern classes to be recognized.

V. Examples

In this section we present some examples of the boundary estimates computed by our algorithm. The images we consider are shown in Figures 10a-e. Figures 10a-c are artificially generated noisy images, while Figure 10d is a FLIR image of a tank and Figure 10e is a satellite image of a cloud.

First consider Figure 10a. It shows a perfect ellipse in an additive Gaussian noise field resulting in a signal to noise ratio $\Delta/\sigma = 1$. Figure 11a shows both the actual elliptical boundary and the estimated boundary when transition probabilities were calculated using the local curvature algorithm. The parameters a and b in equation (10) were 2.0 and 5.0 respectively. For the estimate shown in Figure 11b the transition probabilities were calculated by use of the Kalman filtering algorithm where

$$A = \begin{bmatrix} 1.0 & 0.0209 \\ -0.058 & 1.0 \end{bmatrix}$$

$$b = \begin{bmatrix} -0.6688 \\ 1.856 \end{bmatrix}$$

$$K = \begin{bmatrix} 0.568 & 0.0138 \\ -0.0365 & 0.5683 \end{bmatrix}.$$

As expected, since the Kalman filter approach makes use of more *a priori* information, it generates better estimates. Computationally, it is also more efficient in terms of the number of nodes placed on the graph. The elliptical boundary contains roughly 180 edge elements. Hence, if an algorithm did not expand any extraneous graph nodes it would expand slightly fewer than 180 nodes. The curvature algorithm expanded 307 nodes while the Kalman filter algorithm expanded only 200 nodes.

Next consider Figure 10b. It shows a distorted ellipse whose boundary was generated by use of the dynamical model given in (11) and (12). It also is imbedded in an additive Gaussian noise such that $\Delta/\sigma = 1$. Figure 12a shows the true and estimated boundary using the Kalman filter algorithm, and the state space model parameters which generated the true boundary. For this model

$$A = \begin{bmatrix} 1.0 & 0.0209 \\ -0.058 & 1.0 \end{bmatrix}$$

$$b = \begin{bmatrix} -0.6688 \\ 1.856 \end{bmatrix}$$

$$K = \begin{bmatrix} 0.9285 & 0.0194 \\ -0.0538 & 0.9285 \end{bmatrix}$$

These matrices were generated by choosing $N=180$, $M=3N$, $\theta=90°$, $\rho=3/5$, $c^T = [32,32]$, $\sigma_u=1$ and $\sigma_w=1/\sqrt{12}$. To demonstrate the robustness of the algorithm to poorly estimated model parameters, we perturbed these parameters by roughly 20%. Specifically we chose $N=160$, $\theta=70°$, $\rho=3/4$, $c^T = [26,26]$, $\sigma_u=1$, and $\sigma_w=1/\sqrt{12}$. This results in

$$A = \begin{bmatrix} 1.0 & 0.0317 \\ -0.0489 & 1.0 \end{bmatrix}$$

$$b = \begin{bmatrix} -0.962 \\ 1.2714 \end{bmatrix}$$

$$K = \begin{bmatrix} 0.9285 & 0.0294 \\ -0.0454 & 0.9285 \end{bmatrix}$$

Figure 12b shows the estimated and true boundaries in this case. While the resulting performance in the two cases is very similar, it appears that accuracy of model parameter estimation can substantially influence computation time. With the correct model parameters the algorithm expanded only 190 nodes while with the incorrect parameters the algorithm expanded 438 nodes.

Figure 10c shows a circular object with a rectangular projection of the type discussed in Section III. The signal to noise ratio in this picture is also one. Figure 13 shows the true and estimated boundary for this picture using the curvature algorithm for calculating transition probabilities. In this case, the data swaths used in the likelihood calculations were large enough to assure proper performance.

Finally, Figures 14 and 15 show the results of applying our algorithm to real image data. Figure 14 shows the estimated boundary for the FLIR image of the tank shown in Figure 10d while Figure 15 shows the estimated boundary for the satellite cloud image of Figure 10e.

VI. Comments and Conclusions

By viewing boundary estimation in terms of MAP estimation of a Markov process state sequence, in Section II we presented a maximum likelihood framework to serve as a basis for the design and analysis of sequential boundary estimation algorithms. While optimal estimation required use of all the picture data, computational constraints limited the amount of data which could be practically employed by an algorithm. As a result, the particular suboptimal algorithm presented in Section III only incorporated picture data in swaths about hypothesized boundary edge sequences. The results of Section IV demonstrated the power of the likelihood formulation in doing formal algorithm analysis for the purposes of comparing different algorithms and improving performance of a specific algorithms.

It should be pointed out that the particular algorithm presented in Section III is but one of many possible suboptimal algorithms which are consistent with the general maximum likelihood formulation, and there is still potential for developing algoirthms with both improved performance and reduced computational complexity. Design of such algorithms involves two important steps. One must choose an appropriate Markovian boundary generation model, and one must decide on how to incorporate picture data into the likelihood computations. With this in mind we make some final comments on these subjects.

Boundary Model Design

In choosing a boundary model one must decide upon the type of *a priori* information regarding boundary curvature and shape which is available. This information can be generally classified as local or global in nature. For example smoothness is a local property of a boundary, while closure is a global property. Though the two models we've made use of incorporate both local and global information, the curvature model emphasizes local information while the dynamical system model emphasizes global information. In particular, the curvature model is described in terms of behavior of short sequences of edge elements. It does not make use of the fact that the boundaries of interest are closed. On the other hand, although curvature information is implicitly incorporated into the parameters of the dynamic model, this model was explicitly developed to exploit *a priori* knowledge that the boundaries of interest were closed.

Picture Data Usage

In view of the optimal formulation one is clearly interested in making use of as much picture data as possible. However, to take advantage of the computationally attractive sequential maximization procedures such as A* or dynamic programming, one must limit the use of picture data. We chose to consider (4x4) data windows and hence limited our use of picture data to swaths of roughly 2 pixel width on each side of an hypothesized boundary. Although as demonstrated in the comparison with the Martelli algorithm in Section IV, this window is useful in preventing a first erroneous edge element decision, since different swaths contain different data, performance is degraded in comparison with optimal data usage. As a result, a computationally feasible algorithm which makes use of a larger data window containing a greater number of paths for comparison, could considerably improve performance. Use of a larger window also appears important for images containing large numbers of pixels where grey level changes along boundaries may be more gradual. One approach for sequentially estimating boundaries through larger window blocks is given in [5]. It might also be mentioned that a more detailed discussion of the dependence of boundary error on data usage is given in [1].

In conclusion, the guiding principle in this work has been to formulate an optimal approach to the problem, and then to design a suboptimal realization whose performance can be fairly well understood and analyzed.

References

[1] D. B. Cooper, "Maximum Likelihood Estimation of Markov Process Blob Boundaries in Noisy Images," IEEE Trans. on PAMI, Vol. 1, No. 4, Oct. 1979.

[2] D. B. Cooper and H. Elliott, "A Maximum Likelihood Framework for Boundary Estimation in Noisy Images," Proc. of the IEEE Computer Society Conf. on Pattern Recognition and Image Processing, Chicago, 1978, pp. 25-31.

[3] A. Martelli, "An Application of Heuristic Search Methods to Edge and Contour Detection," Communications of the ACM, Vol. 19, No. 2, February, 1976, pp. 73-83.

[4] G. D. Forney, Jr., "The Viterbi Algorithm," Proc. IEEE, 61 March, 1973, pp. 268-278.

[5] L. L. Scharf and H. Elliott, "Aspects of Dynamic Programming in Signal and Image Processing," Colorado State Univ. Technical Report No. DEC79-C, Dec. 1979 (Submitted IEEE Trans. on Aut. Cont.).

[6] N. Nilsson, Problem-Solving Methods in Artificial Intelligence, McGraw-Hill, New York, 1971, pp. 54-70.

[7] N. E. Nahi and S. Lopez-Mora, "Estimation-Detection of Object Boundaries in Noisy Images," IEEE Trans. on Automatic Control, Vol. AC-23, No. 5, Oct. 78, pp. 834-845.

[8] N. E. Nahi and M. H. Jahanshahi, "Image Boundary Estimation," IEEE Trans. on Computers, Vol. C-26, No. 8, Aug. 77, pp. 772-781.

[9] M. H. Hueckel, "A Local Visual Operator which Recognizes Edges and Lines," Journal of the ACM, Vol. 20, No. 4, October, 1973.

[10] H. Elliott, D. B. Cooper, P. Symosek, "Implementation, Interpretation and Analysis of a Suboptimal Boundary Finding Algorithm," Proc. IEEE Computer Soc. Conf. on Patt. Rec. and Image Processing, Chicago, 1979.

[11] F. Cohen, D. B. Cooper, H. Elliott, P. Symosek, "Two-Dimensional Image Boundary Estimation by Use of Likelihood Maximization by and Kalman Filtering," Proceedings of Int. Conf. on Acoustics, Speech, and Signal Processing, Denver, April, 1980.

[12] L. Reiss and D. B. Cooper, "The Ripple Filter: An Algorithm for Region Growing in Scene Analysis," Proc. Conf. on Computer Software and Applications, Chicago, Nov., 1979.

ACKNOWLEDGMENTS

This work was partially supported by the U.S. Army Research Office under grant DAAG29-78-G-0124, by the National Science Foundation under grant ENG77-26788, and by the Office of Naval Research under grant N00014-75-C-0518.

The authors would also like to thank Fred Hansen of Colorado State University for helping with some of the computer programming on this project.

Martelli Algorithm

		a	b	c	d	e	f	g	h
1 correct –	ρ	2.24	1.34	1.155	1.34	1.50	1.50	1.875	1.789
2 incorrect	P_ρ	.0125	.090	.125	.090	.067	.067	.031	.037
1 incorrect –	ρ	−.447	.447	.577	.477	.250	.250	−.125	0
2 correct	P_ρ	.673	.327	.281	.327	.402	.402	.548	.500
Average	P_a	.342	.2085	.203	.2085	.2345	.2345	.289	.268

Suboptimal Algorithm

		a	b	c	d	e	f	g	h
Either 1 or	ρ	.667	.756	.816	.756	.756	.756	.756	.707
2 correct	P_ρ, P_a	.252	.224	.207	.224	.224	.224	.224	.239

Optimal Algorithm

		a	b	c	d	e	f	g	h
Either 1 or	ρ	1	1	1	1	1	1	1	1
2 correct	P_ρ, P_a	.159	.159	.159	.159	.159	.159	.159	.159

Table 1 - Comparison of Martelli, Suboptimal, and Optimal Algorithms

Case	Algorithm	N	Δ/σ	ρ_{ij}	P_{ij}
1	1	4	1	1.414	.078
			2	2.828	.002
		8	1	2.0	.023
			2	4.0	.000
	2	4	1	1	.159
			2	2	.023
		8	1	1.414	.078
			2	2.828	.002
	3	4	1	.577	.281
			2	1.632	.052
		8	1	.816	.209
			2	2.309	.011
2	1	4	1	1.414	.078
			2	2.828	.002
		8	1	2.0	.023
			2	4.0	.000
	2	4	1	.816	.209
			2	1.632	.052
		8	1	1.154	.125
			2	2.309	.011
	3	4	1	.632	.266
			2	1.705	.045
		8	1	.894	.187
			2	2.411	.008

Table 2 - Comparison of Algorithms 1, 2, and 3.

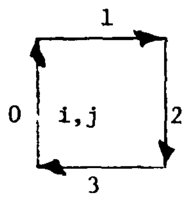


Figure 1a. Four directed edge elements defined by an arbitrary pixel (i,j) and their corresponding direction numbers d.

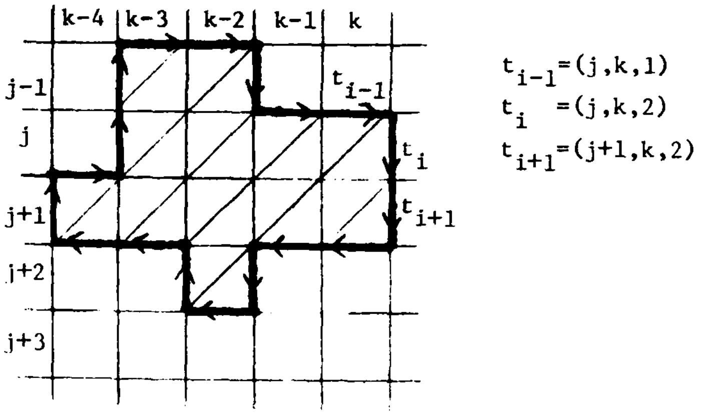


Figure 1b. Example of a closed directed boundary.

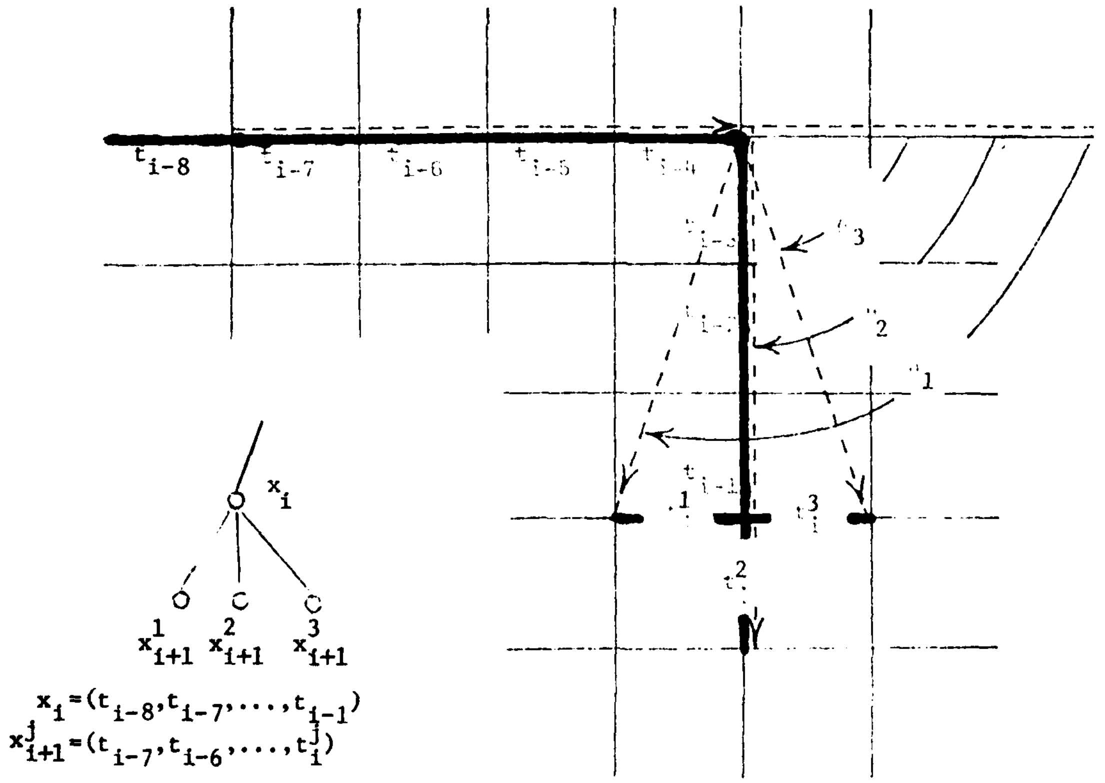


Figure 2. Relation between boundary edge sequences and graph nodes (Markov states).

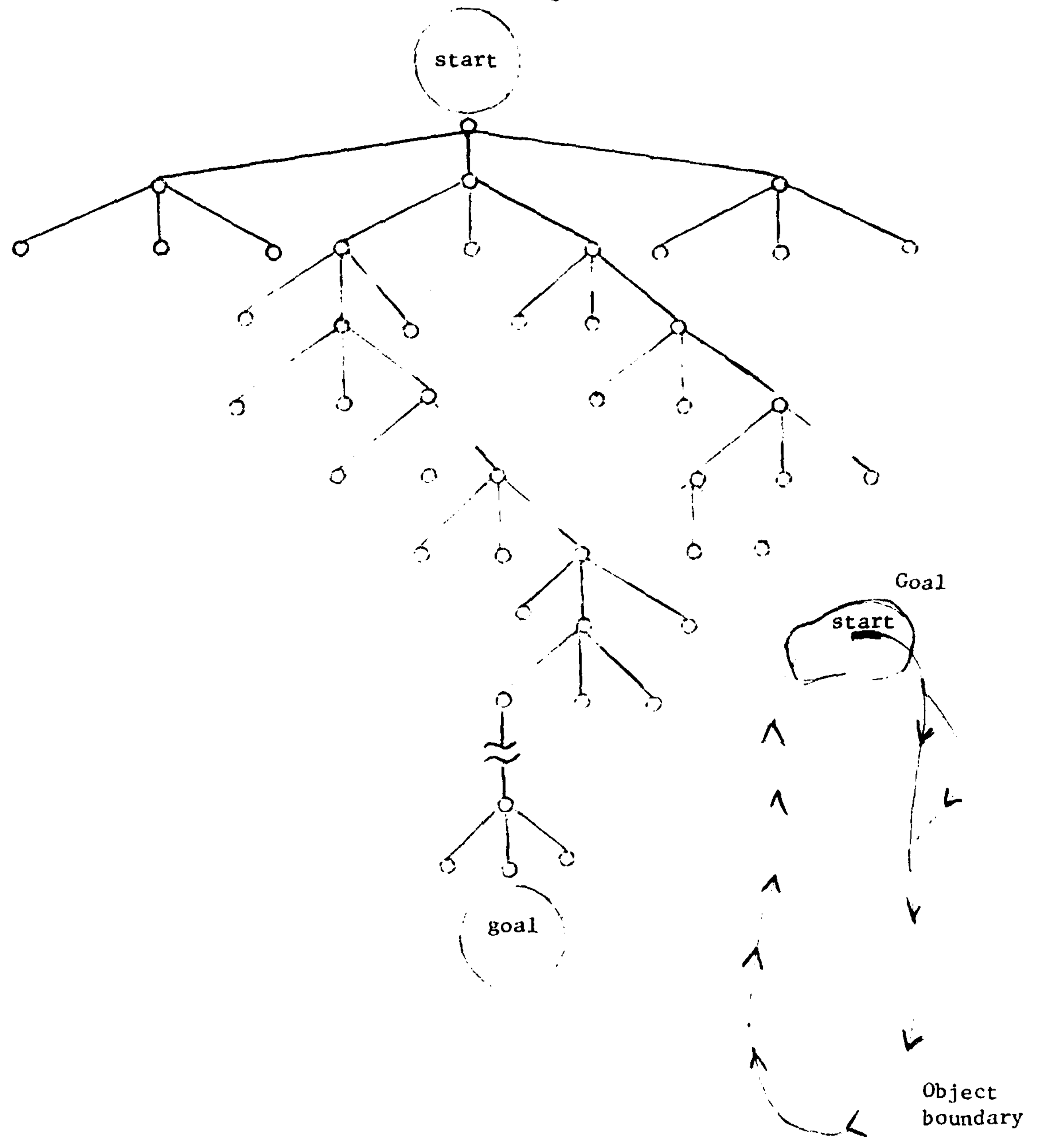


Figure 3. Graph structure corresponding to two alternative object boundaries.

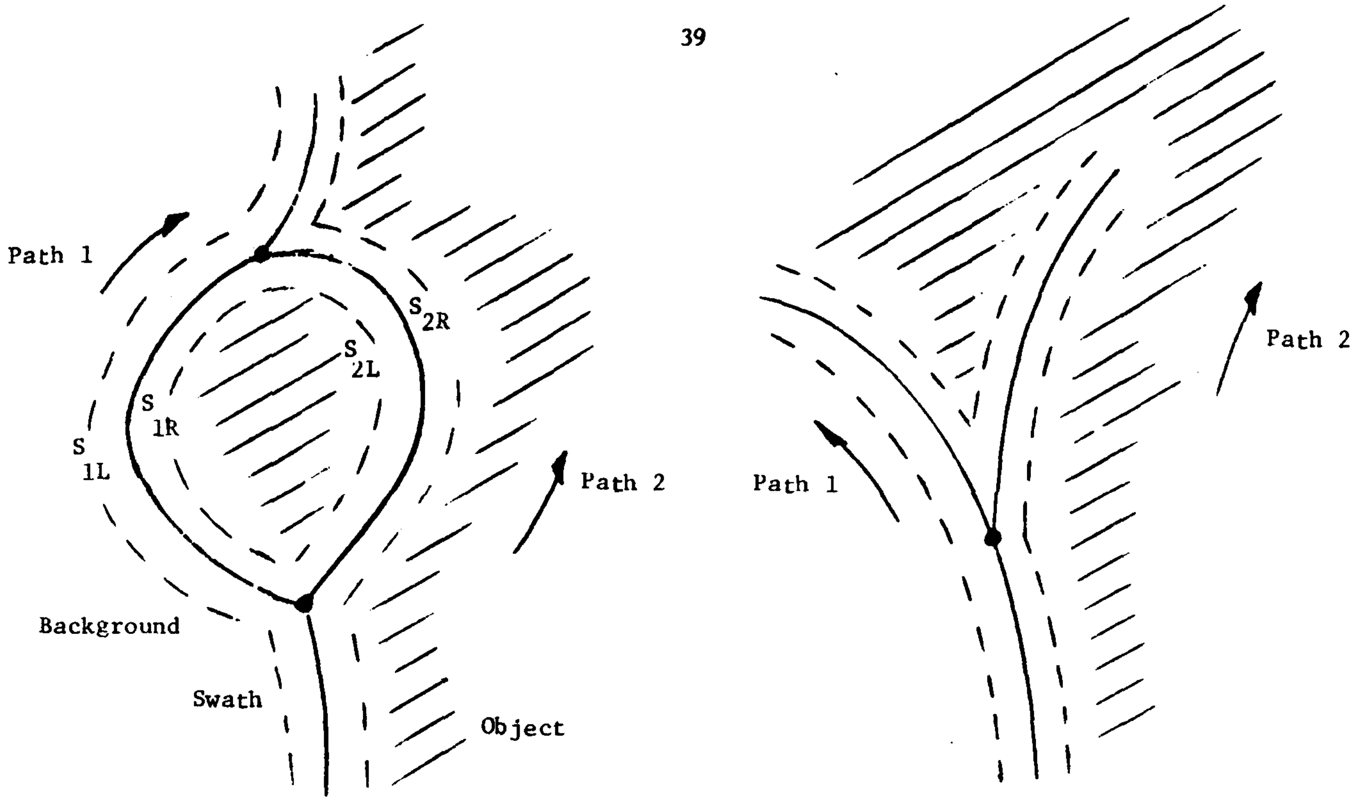


Fig. 4a Examples of alternative boundary path segments with disjoint data swaths.

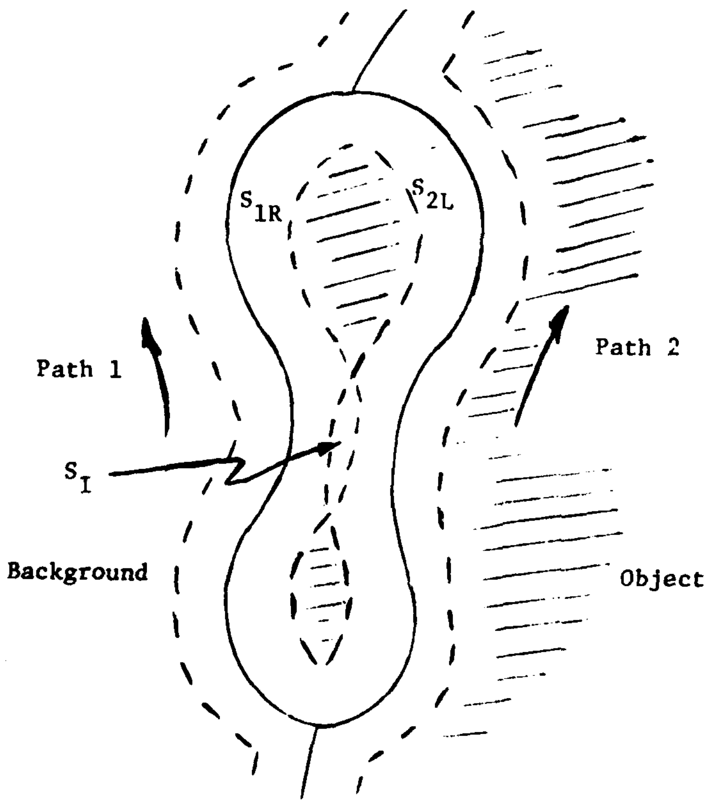


Fig. 4b Example of paths with overlapping data swaths.

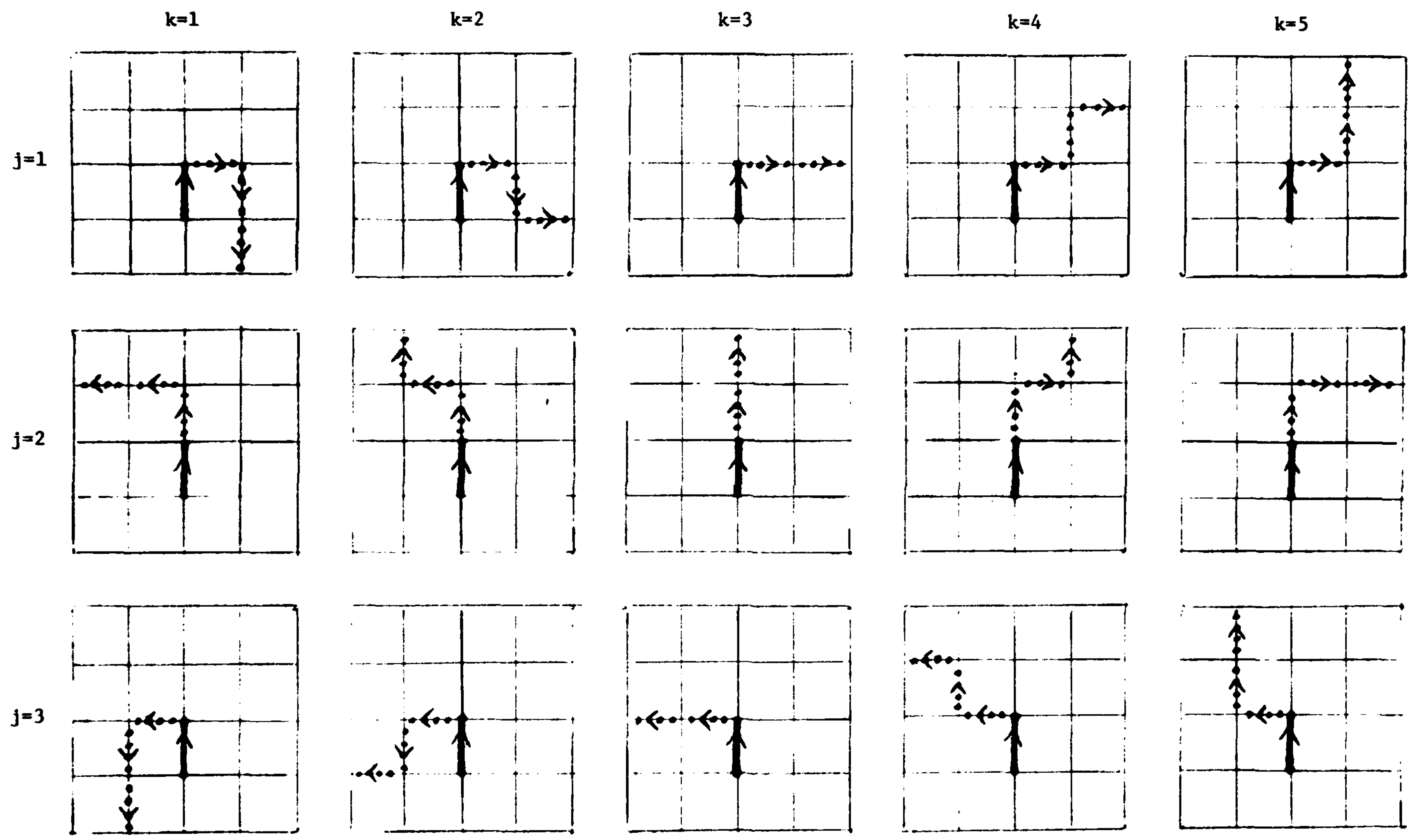


Figure 5. Extension paths for calculating data likelihoods of successor nodes x^j_{i+1}, j=1,2,3.

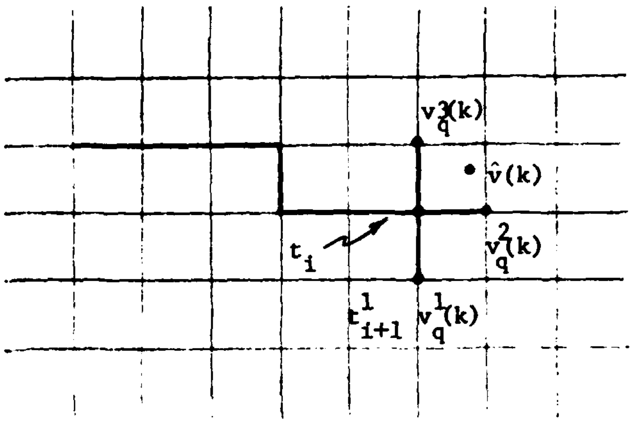


Figure 6. Kalman filter estimate $\hat{v}(k)$ and corresponding quantized data points $v_q(k)$.

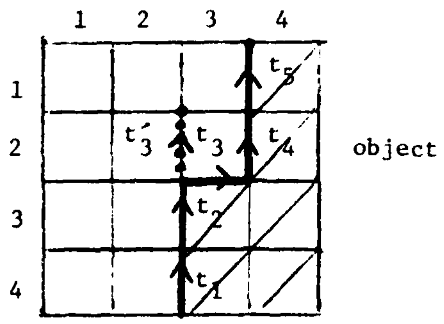


Figure 7. (4x4) section of a picture.

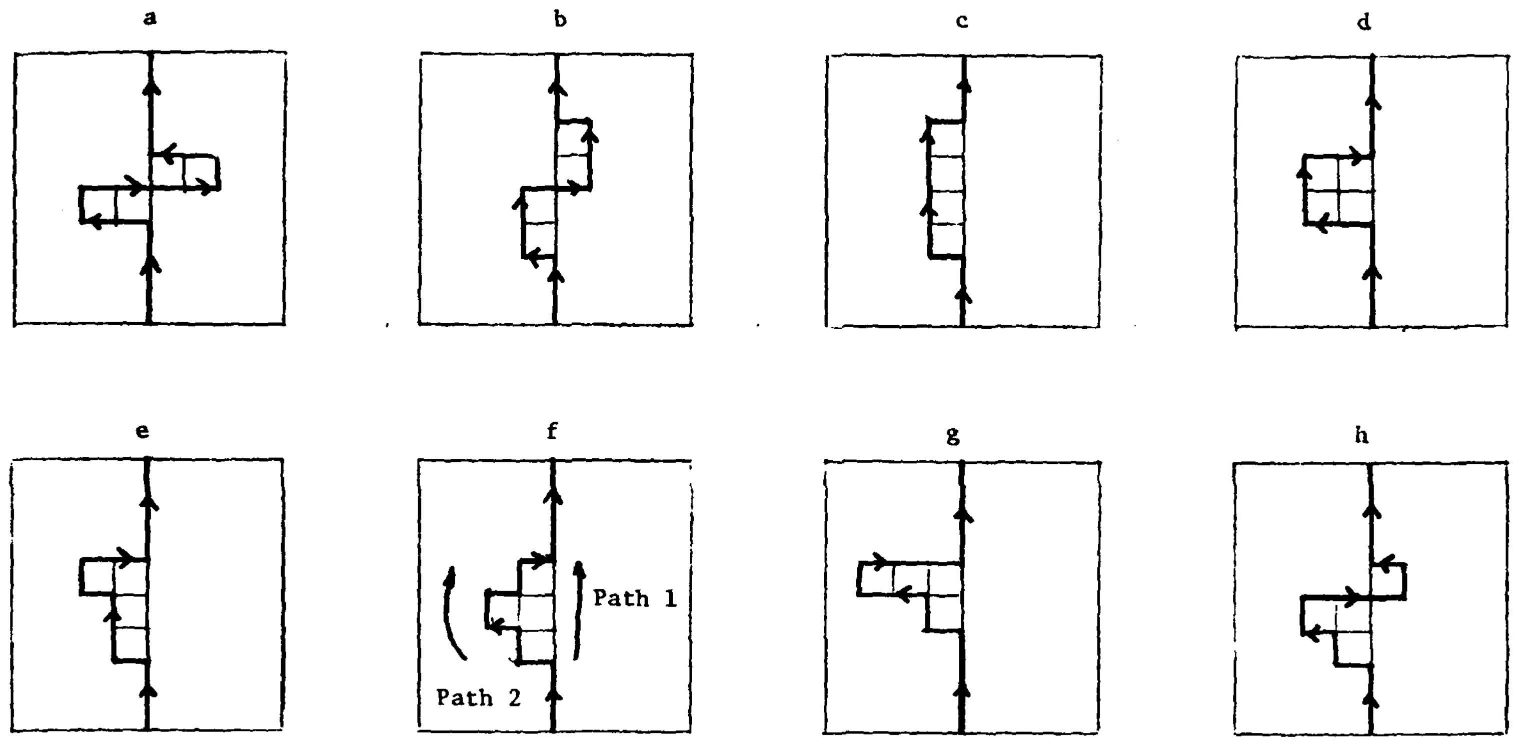


Figure 8. Eight pairs of paths, each differing by four pixels. Path 1 is straight and Path 2 is curved.

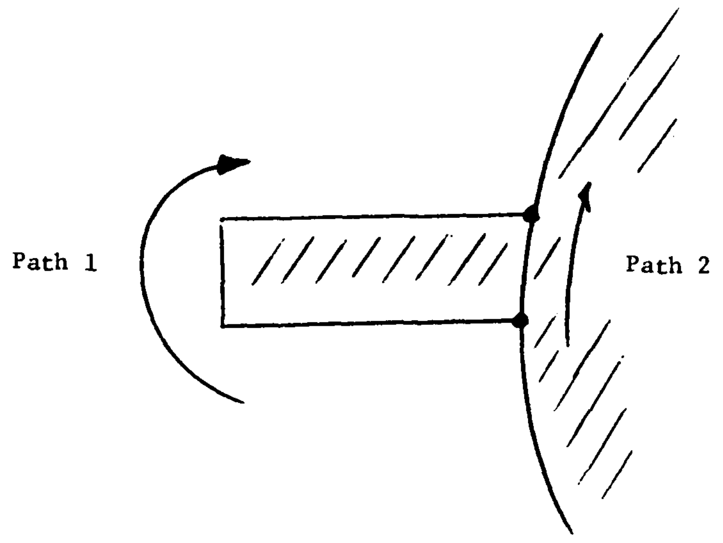


Fig. 9 Boundary path segment containing a projection.

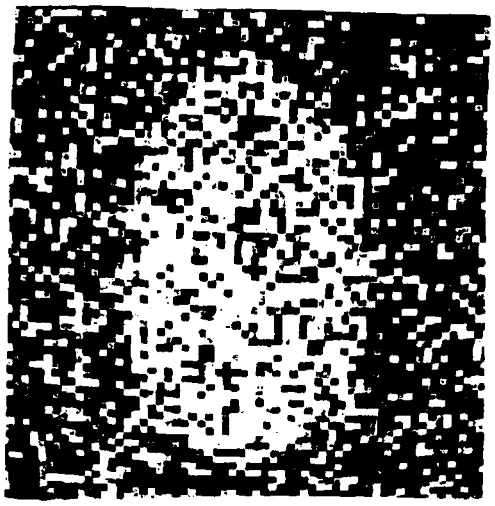


Figure 10a - Perfect Ellipse, $\Delta/\sigma=1$.

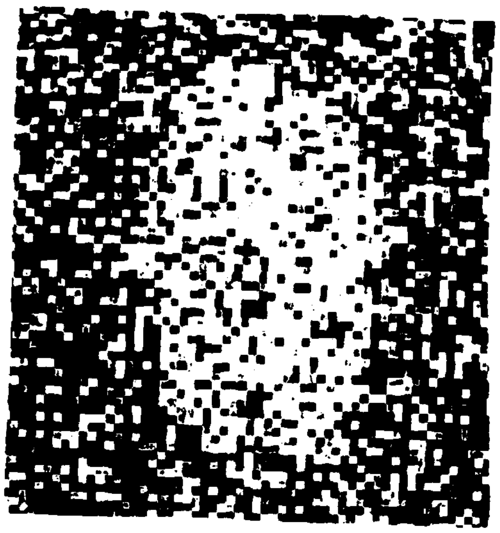


Figure 10b - Distorted Ellipse, $\Delta/\sigma=1$.

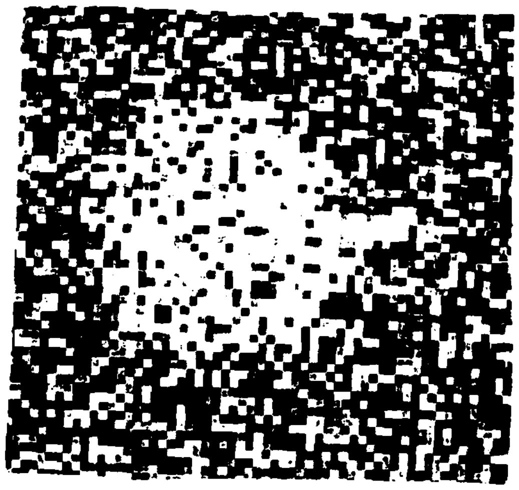


Figure 10c - Circle and Projection, $\Delta/\sigma=1$.

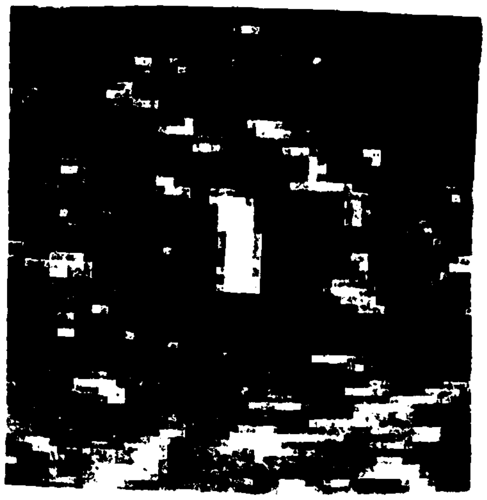


Figure 10d - FLIR Image of a Tank

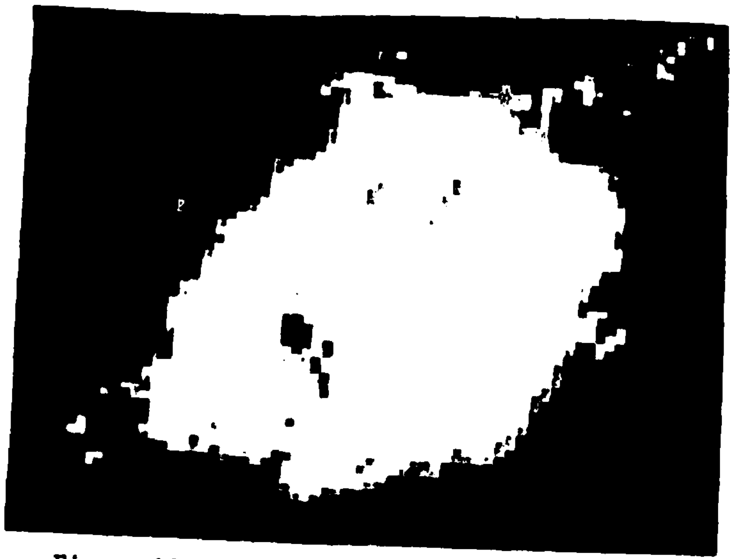


Figure 10e - Satellite Image of a Cloud.

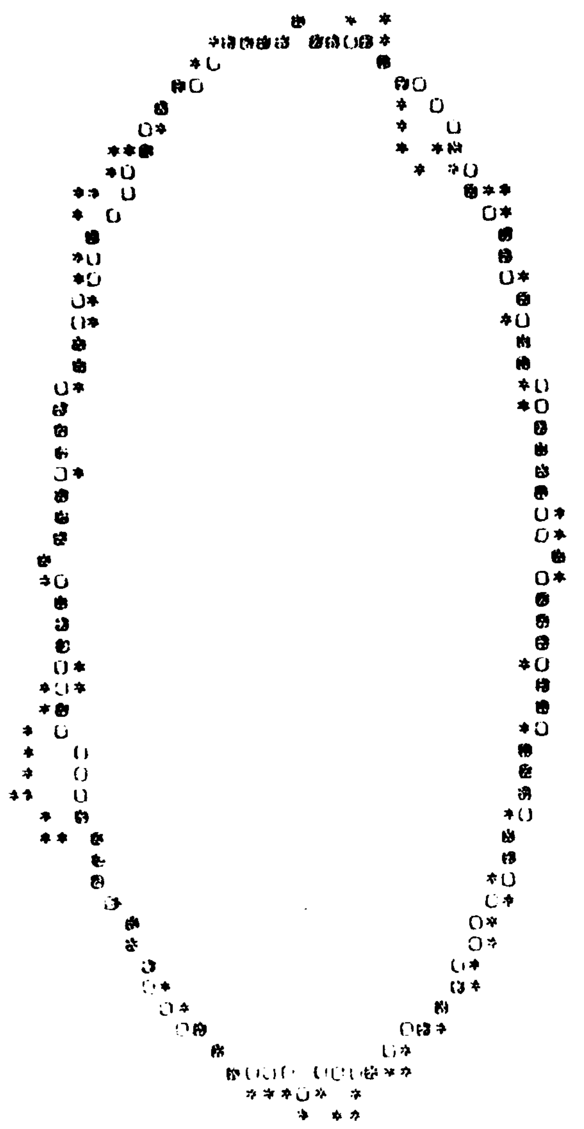


Figure 11a - True (0) and estimated (*) boundaries for perfect ellipse using curvature algorithm.

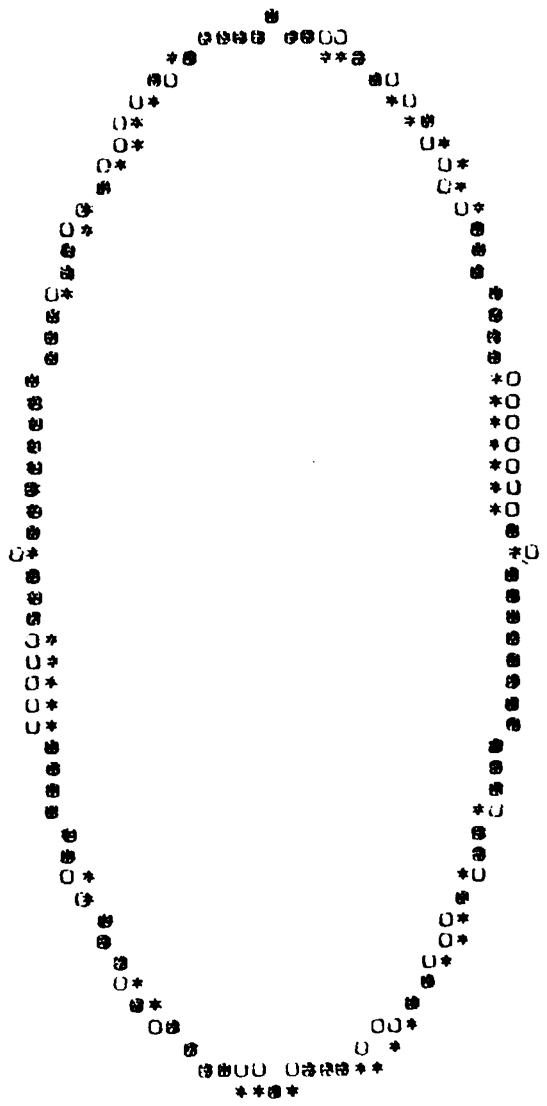


Figure 11b - True (0) and estimated (*) boundaries for perfect ellipse using Kalman filter algorithm.

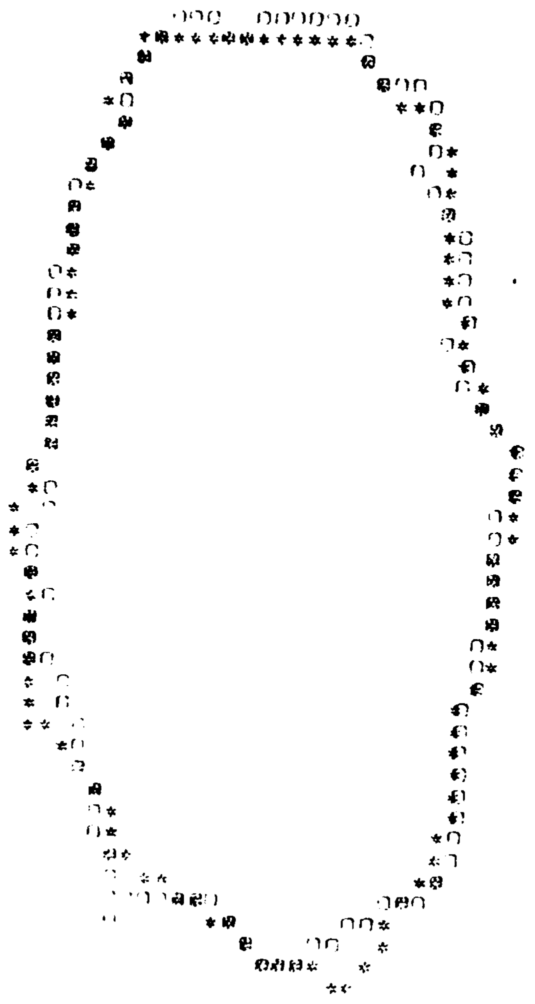


Figure 12a - True (O) and estimated (*) boundaries for distorted ellipse using Kalman filter algorithm and correct model parameters.

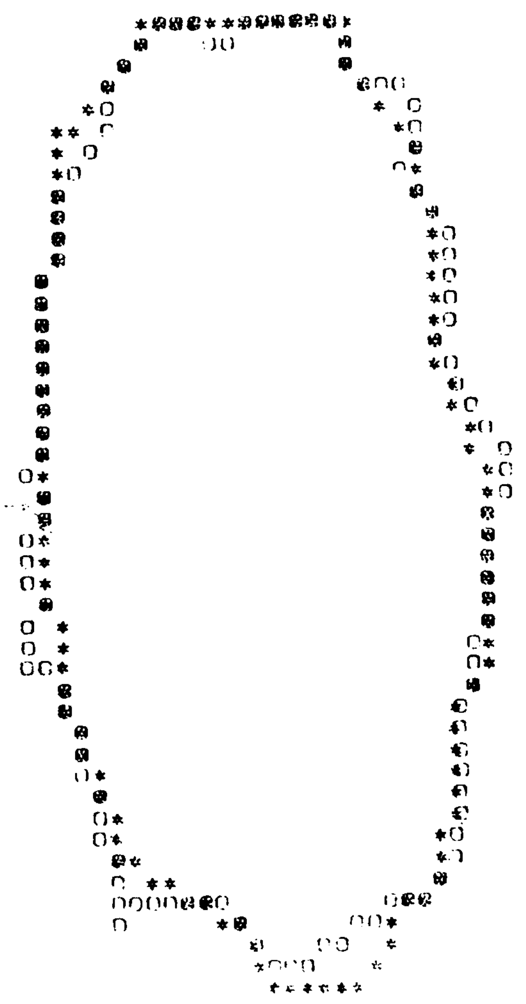


Figure 12b - True (O) and estimated (*) boundaries for distorted ellipse using Kalman filter algorithm and incorrect model parameters.

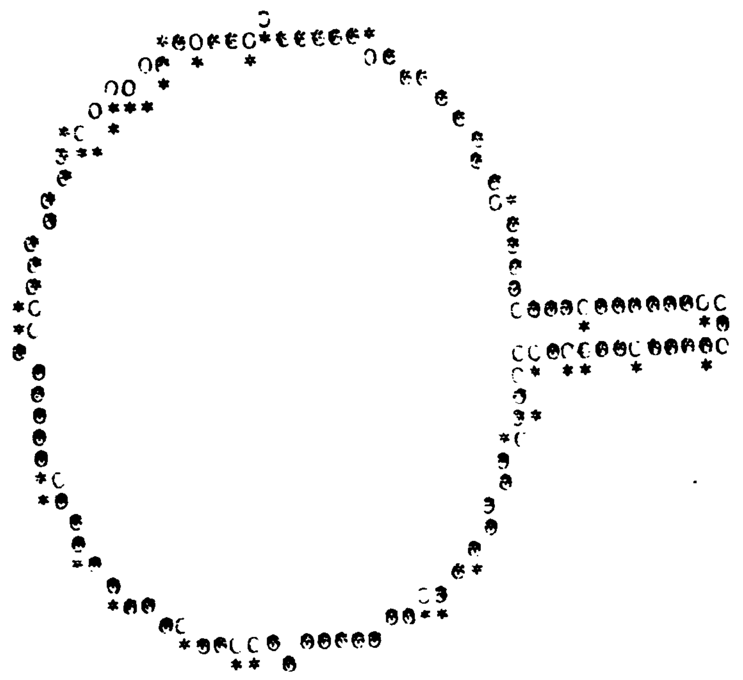


Figure 13 - True (0) and estimated (*) boundaries for circle and projection using curvature algorithm.

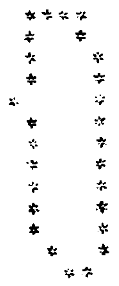


Figure 14 - Estimated tank boundary.

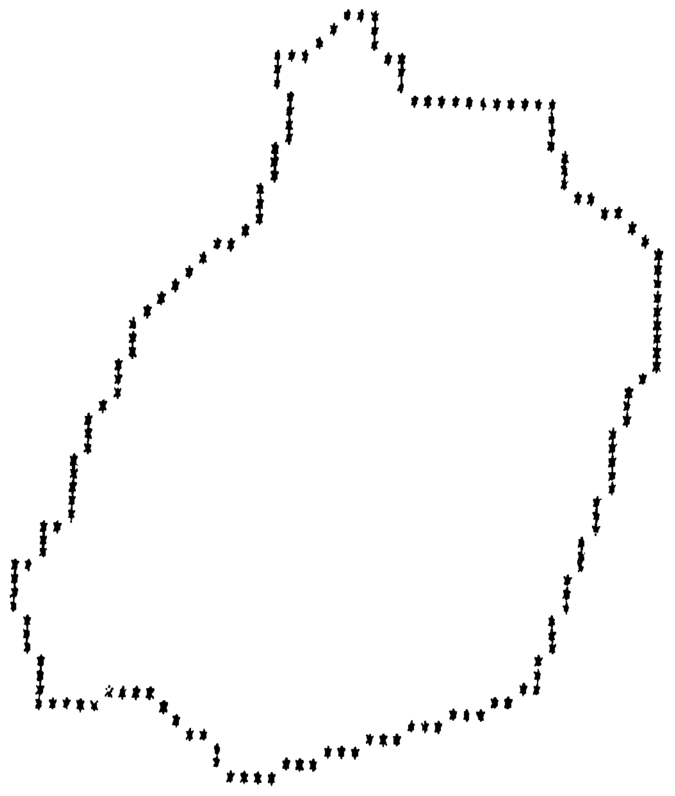


Figure 15 - Estimated Cloud Boundary